described using XML notation in Fig. 3. For any custom validators that are found, the custom validation is executed. This effectively allows plugging and unplugging of validation code. Validators can be added or removed using the addValidator or removeValidator methods. The "validate" method serves as an entry point to this code, and is invoked according to the developer's code hooks, as discussed earlier.

In prior art approaches, developers typically rely on the native capabilities of each widget. For example, when using an interactive development environment such as VisualAge® from IBM, a panel designer creates a panel layout and specifies error checking on the widget itself. ("VisualAge" is a registered trademark of IBM.) An alternative prior art approach is to hard-code the error checking in-line into an application, as discussed earlier. This error-checking code becomes a fixed, static part of the application, and any changes to the data validation require changing the application itself. As is known in the art, changes of this type may become quite complex and may be quite time-consuming, tedious, and error-prone for large applications. Using the techniques of the present invention, on the other hand, the validation is isolated with the data model, and changes thereto may be made more easily and quickly and do not require changing the application itself.

The disclosed techniques enable a consumer of information to fire a validation without concern over what that validation does. That is, the consumer does not need to contain data-dependent validation for data that it may receive from another source. Instead, the data includes its own validation, because the data and validation are packaged together. Furthermore, the

consumer can mutate the data, and the self-contained validation will ensure that the result remains valid (or that the mutation is prevented).

The techniques of the present invention may be used to provide support for "VariableModel" class which was described in commonly-assigned U. S. Patent _____ (serial number 09/669,227, filed 09/25/2000), titled "Object Model and Framework for Installation of Software Packages Using JavaBeans™". This patent is hereby incorporated herein by reference as if set forth fully.

As will be appreciated by one of skill in the art, embodiments of the present invention may be provided as methods, systems, or computer program products. Accordingly, the present invention may take the form of an entirely hardware embodiment, an entirely software embodiment or an embodiment combining software and hardware aspects. Furthermore, the present invention may take the form of a computer program product which is embodied on one or more computer-usable storage media (including, but not limited to, disk storage, CD-ROM, optical storage, and so forth) having computer-usable program code embodied therein.

The present invention has been described with reference to flow diagrams and/or block diagrams of methods, apparatus (systems) and computer program products according to embodiments of the invention. It will be understood that each flow and/or block of the flow diagrams and/or block diagrams, and combinations of flows and/or blocks in the flow diagrams and/or block diagrams, can be implemented by computer program instructions. These computer

program instructions may be provided to a processor of a general purpose computer, special purpose computer, embedded processor or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions specified in the flow diagram flow or flows and/or block diagram block or blocks.

These computer program instructions may also be stored in a computer-readable memory that can direct a computer or other programmable data processing apparatus to function in a particular manner, such that the instructions stored in the computer-readable memory produce an article of manufacture including instruction means which implement the function specified in the flow diagram flow or flows and/or block diagram block or blocks.

The computer program instructions may also be loaded onto a computer or other programmable data processing apparatus to cause a series of operational steps to be performed on the computer or other programmable apparatus to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide steps for implementing the functions specified in the flow diagram flow or flows and/or block diagram block or blocks.

While the preferred embodiments of the present invention have been described, additional variations and modifications in those embodiments may occur to those skilled in the art once they learn of the basic inventive concepts. Therefore, it is intended that the appended claims shall be